Prior Art

Description

Field of the Invention

This invention relates to communications networks, and more particularly to automated path determination across multiple nodes and circuits in a network that are constrained by routing requirements and the characteristics or the transmission facilities and switching systems that make up the network. Networks as defined for this application are any interconnections of data transmission facilities and terminal equipment that is capable of joining multiple transmission segments into a transmission path that acts as integral communication circuit and maintains the integrity of the data that is transmitted via that path. The network thus defined is not limited to a specific type of modulation technology, communication methodology, or family of communications protocols. The invention may be embodied in an Operations Supervisory System, a provisioning Application, a soft switch application, a route srever, a switching control system, or embedded in the switching equipment itself..

Although the primary environment for the invention is data communications, it has applications to any mesh of nodes and links that transport something from a location to another location, where there are multiple links and/or nodes to transit to get from the

source to the destination. Examples are networks for transporting natural gas, oil, and electricity. Highways and electronic circuit boards can be modeled as such networks.

Prior Art

Asynchronous Transfer Mode's PNNI routing methodology (within the public domain) provides hierarchal routing as does OSPF for IETF TCP/IP protocols (within the public domain). Both use a Shortest Path First method of selecting paths. Both uses are designed for scaling networks or portion's of networks into suitable sized pieces for an automated routing application to return routes in reasonable timeframes. In both cases, sub-sets of a network are summarized into a simulated node for the use of higher level routing levels. This invention does not claim exclusivity to multiple level networks. It does claim exclusivity to multiple levels where each level has distinct routing properties and that lower levels are summarized as links to the higher levels and that a switching node may appear in multiple levels of the network concurrently. Assigning levels to specific routing properties, allows the method to be useful for resolving over constrained routing requirements. OSPF stipulates the use of Djikstra's algorithm to find routes. PNNI doesn't require it but uses it in its reference implementation and all commercial versions known to the author use it. 5999517 illustrates the use of hierarchy for scaling and also in differentiation show the difference in the inventions use for resolving multiple constraints. 5805593 illustrates a non-Djikstra path determination method, that unfortunately is a well known maze runner method that has been in the public domain since first described in automata articles in the 1950s.

Djikstra's algorithm uses a single metric to determine a route and the way it deals with its candidate list (internal component of the algorithm) and terminates its search, it is incapable of using multiple metrics concurrently. The only known algorithm that uses multiple metrics concurrently that can be found in the patents libraries is 5521910. It does so while building a spanning tree as stated in its claim 1. 5732192 uses multiple metrics during path determination for analyzing the flow of power in an electrical power grid.

The invention doesn't claim exclusivity for the use of multiple metrics, it claims exclusivity to their use in multiple level networks where the inter-relations between levels is based upon the use of multiple metrics for each level and cross level routing decisions are based upon having the information from multiple levels already calculated and available. This avoids a lot of wasted processing in what ATM calls 'crankback' when one of the routing requirements has been violated while exploring a potential path. 5317562 illustrates the state of the art in metrics. It does a path determination on a single metric and then checks that the path meets the route requirements of remaining metrics. If the path fails another metric it is thrown away, the algorithm is biased to not repeat that path, and then it is run again to get another path to consider. This invention calculates all metrics in parallel and throws away any potential paths that fail the route requirements as they are encountered in the path search. Thus all paths returned automatically meet the route requirements. 5893081 illustrates the use of multiple metrics for evaluating paths in automobile routing systems.

Djikstra's algorithm behaves weakly in dense mesh topologies because it is based upon spanning trees that ignore many of the links in a dense mesh to eliminate the redundant paths that may cause loops in the algorithm. This in turn congests the paths that participate in the sparse spanning tree. The redundant paths and their potential for spreading congestion across themselves is the primary motivation for Dense Mesh deployment in networks.

There is much literature in the public domain that discusses routing in Dense Mesh networks. Most of it is in the area of Operations Research on how to break the problem down into subsets using summarization of regions of the network. None has been found that deals with topologies resulting from large numbers of links that share common termination points which is the primary impetus of Dense Mesh networks in fiber optic networks with large numbers of wavelengths (i.e. lambdas) per fiber and large numbers of fibers per cable bundle. This claim alone is a major contributor to the good performance of the method. The second best contributor is the heuristic called the pheromone aggregate that is borrowed from the behavior of intelligent autonomous agents that in turn borrowed it from insect behavior (Bees and Ants).5937397 illustrates the use of heuristics roughly similar to how pheromone aggregates do so. 6151327 and 6016485 discusses summarization of the network to reduce route determination time.

6205154 illustrates the state of the art in provisioning optical networks. It explains how to match tail paths at subsidiary rates to groups of open sub-channels in backbone links that

already exist between a source and destination switching center. Circuit path determination in optical networks is typically a manual process.

6205154 illustrates the state of the art in resource allocation in that each resource is allocated in isolation from the other resources needed to establish a circuit rather than in a coordinated way as the invention does.

6016306 illustrates a weight metric that is similar to the minimum type metrics of the invention. They differ in that the minimum type considers all nodes and or links along a path rather than just key choke points.

Jargon

Lambda – Synonymous with Wavelength – It is a specific portion of the light spectra. From physics where the Greek letter lambda is uniformly used as the symbol for the frequency of a light.

MEM – Micro electronically controlled Mirrors – A new emerging technology for redirecting light

O-E-O – Optical to Electrical to Optical devices such as SONET switches

WDM – Wave Division Multiplexing where multiple light sources all using a different light wavelength (i.e. lambda) to achieve multiple data streams over single fiber.

Background of the Invention

Today we are seeing a paradigm shift in how data networks are constructed. Existing networks backbones composed of fiber optic media for data transport, SONET framing, and electronic switching. Fibers carry a single data stream at data rates of 2.5 Gigabits per second and the aggregate switching capacity in switching centers is in the order of magnitude of 100-200 Gigabits. Fiber Optic signals need to be regenerated every 5-10 kilometers with expensive regeneration equipment that converts the optical signals to electronic signals that regenerate the digital signal and then amplify the signal for retransmission. End to end circuits are planned, engineered, and installed by highly trained personnel using manual methods and exotic test equipment in the switching centers.

Overlay telephone and data networks that utilize the backbone optical network use adaptive routing technologies such as ATM's PNNI, TCP/IP's OSPF with the switches making dynamic switching with no manual intervention. Intermediate SONET networks use static engineered rings with manual provisioning and automated failover or recovery switching.

Optical pumping and newer fiber optics technology has extended the data regeneration distances to 50 or more kilometers. Transmission equipment using Dense Wave Division Multiplexing are transmitting 128 or more data streams per fiber at data rates of 10 Gigabits and with laboratory systems doing 256 or more data streams per fiber at data

rates of 40 or 80 Gigabits. The aggregate data rates traversing switching centers are now 1000s of terabits far exceeding the aggregate switching capacity of the electronic switching equipment. New switching technology using micro-mirrors or bubbles support switching optical data streams without optical conversion. They are capable of having their switching remotely controlled However, they inject additional signal degradation without providing signal regeneration. Discrete signal regenerators using tunable VCSEL laser transmitters are working in the lab and can provide signal regeneration and translation between wavelength (lambda) configurations. These units are expensive and will only be used where they are required to by the end to end data circuits.

Switching centers are already seeing a diversity of DWDM configurations and switching technologies, each presenting their own constraints to routing.

Constraints are:

1.) When insertion losses aggregate from fiber and or switch traversals reach threshold values, signal regenerators need to be installed or switched into the path.
2.) When DWDM configurations of one fiber don't match those of the next fiber to be transited along a data path, wavelength (i.e. lambda) translators must be installed or switched into the data path.
3.) Mirrors preserve wavelengths from end to end, so paths that use Lambda switching as their only switching medium need to have the same wavelength assigned to a path from end to end. This introduces the constraint that to create a multi-hop path we need to know which wavelengths are available for assignment

for all fibers that participate in the path so that we can choose one that is available for all fibers. There is a possibility of lambda (i.e. wavelength) assignment blocking. So if this is not done as part of path determination, we can wind up in a situation of endless crankback.

4.) When a wavelength must have a lower data rate signal added or dropped for delivery along a secondary path, this requires either conversion of the wavelength to an electronic signal for electronic switching or special optical add/drop units. The electronic switching has a limited amount bandwidth and this adds constraints on its use. Optical add/drop units add significant insertion loss and affect the need for signal generators.

5.) O-E-O conversion adds latency and jitter to circuits that use them.

6.) O-E-O switching adds even greater amounts of latency and jitter to circuits that use them.

There is a strong need to provide the same kind of automated path determination and dynamic path establishment for optical networks as there are for TCP/IP and ATM networks. This invention provides a method for the first of those needs.

Summary of the Invention

The General Invention provides a framework of using multiple hierarchal routing levels to resolve routing constraints and provide path segments that can be used by higher levels to construct paths using those segments as simple links. This allows the higher levels to

deal only with those constraints that are applicable to their defined properties. It also groups all the possible alternative paths between 2 points into a small number of "logical links" thus reducing the processing requirements of the route exploration phase and deferring the complex computations for final path commitment to a point where only a small subset of the potential possibilities are being explored.

The specific Invention provides route determination for optical core dense mesh networks that support DWDM over optical fibers at data rates of 2.5 Gigabytes, 10 Gigabytes, 40 Gigabytes, and higher data rates.. It supports switching systems that include a mix of electronic and pure optical switching (micro-mirrors, bubbles, or equivalents), discrete data regeneration devices, and discrete lambda translation devices.

The Layer definitions are,

a.) Multiplexor switching that supports mapping STS1, STS3, STS12, and STS48 channels into lambda data streams at STS48, STS192, and STS768 data rates.

b.) OEO switching that provides grooming for channel multiplexing, lambda translation, and data regeneration

c.) Lambda Translation using discrete lambda translation units

d.) Data Regeneration using discrete regeneration units

e.) Lambda switching using micro-mirror and bubble switching units

f.) Whole fiber switching with Optical Add/Drop Multiplexing

Brief Description of the Drawings

All of the figures are embedded in the text at appropriate locations. They are for explanatory purposes only and are not meant to convey scale or other physical dimensions. The sole use of the drawings are to provide tutorial assistance for the text descriptions.

Detailed Description

The invention is an algorithm that provides optimal paths through a mesh of nodes and links to provide the optimal "simplest shortest path first" that meets the requested routing requirements. The algorithm has a built in model of the mesh and decomposes the mesh into multiple levels to meet the complex routing requirements.

The best embodiment of the invention is a software object that we will call the "GRAPH" that can be integrated into a provisioning system, an automated switching controller, or in a distributed variation embedded in the switching control logic of switches.

A key characteristic of routes that are returned by the Graph object is that it will prefer to return the simplest routes before it returns more complex routes and when alternative routes are of the same simplicity, it will return the shorter of the simplest possible route. The general definition of "simplicity" is the route that can be completed at the lowest possible level in the Graph. This is a somewhat academic answer. Let's use an example

from optical networking with DWDM and a diversity of switching methodologies. The lowest level supports whole fiber switching using micro-mirror (or equivalent) switching. Where all lambdas of a fiber are switched as a group. The next lowest level is switching individual lambda(s) using the same.micro-mirror switching. The next higher layer introduces lambda specific signal regeneration (clocking and amplitude) where needed. The next higher level introduces lambda translation only when necessary. The next level introduces O-E-O switching where necessary. The next level above that introduces add/drop multiplexing where necessary. The simplest possible circuit is one that doesn't require signal regeneration, lambda translation, O-E-O switching, or add/drop multiplexing. The next more complex route requires the fewest signal regenerators along its path. The next most complex route requires the fewest lambda translations along its path, ad infinitum.

The simplest route is practically motivated; it is the cheapest route possible. Data regeneration is expensive. At the time this disclosure is written the normal method is to use an O-E-O switch in the path. The invention anticipates discrete single lambda signal regenerators and lambda translators (signal regenerators that can change their output wavelength to whatever is required). These will be cheaper than using O-E-O switching for that function but they will still be more expensive a simple circuit.

The Graph object accomplishes "Simplest Shortest First" by routing in discrete levels, where the lowest (i.e. the simplest level) provides the links used by the higher (i.e. more complex levels). If it is possible to complete a route between A and B at the lowest level,

then requests at the higher levels will return a simple path. Multi-hop routes at higher levels will see the simple route between A and B as a single hop and will prefer them to other more complex paths.

Another characteristic of Graph is that it deals with Dense Mesh Networks where there are large numbers of alternative paths between neighbors and many alternative neighbors to choose between for a path that will lead to the destination. The Graph deals with this by using a 2 step approach. Its routing cache contains lists of "approximate paths" that all lead from A to B and are ordered using metrics that support "simplest, shortest, first". It selects an "approximate path" and "analyzes it" for possible selection. If it meets the selection test, it is then committed, and the exact path chosen is then removed from the set of 'approximate paths" in the cache. When the set of "approximate paths" is exhausted or it "ages out", the cache fill algorithm for the particular level is run to refill it. If it can't find simple paths like it did previously, it will fill itself with more complex paths if that is possible. On the alternative side, if enough simple paths are surrendered and a critical threshold is passed, cache fill is triggered in anticipation of filling with simpler paths Cache fill at higher levels will prefer to fill from lower levels Only when they can't will it fill by running route exploration at its own level.

At the lambda switching level, a cache entry will be a list of fibers and lambda assignments for those fibers that interconnect 2 nodes. We group all the fibers (and their lambdas) that have similar DWDM configurations that terminate in the same pairs of nodes into a single logical link that during route exploration we can treat as a single link.

This dramatically reduces the amount of computations required for route exploration (that is where most Dense Mesh routing algorithms fail). We segregate logical links with differing DWDM configurations (differing numbers of channels per fiber or maximum clocking rates per lambda primarily) by fragmenting the cache into multiple entries; one for each DWDM configuration. This allows route exploration to focus on the richness of the inter-connection of neighbors without being swamped with repeated computations for almost identical link traversals.

Higher-level logical links are composed of the cache entries of lower levels. Cache entries of higher levels prefer to use existing cache entries of lower levels where possible. If the lower cache level entry exists and their numbers of "approximate paths" exceed a configuration threshold, path exploration between these nodes will not occur at this level. Only when the lower level cache cannot provide enough "approximate paths" will we have a cache entry that combines both lower level cache entries and routes explored at this level.

We will present a simplified example using both of the above principles. It will show a portion of a hypothetical network at the raw lambda and micro-mirror level and at the lambda translation level. We will for this example ignore the signal regeneration level to keep the example as simple as possible.

Figure 1 shows a 3 x 3 portion of a larger network. Each node is a circle that has a switching or interconnection matrix that switches individual lambdas while preserving

their optical integrity (Micro-mirror and bubble switching have these capabilities.). The interconnections represent bundles of DWDM connections that terminate at the same nodes. The thicker lines on the left side represent 128 lambda DWDM configurations and the skinnier lines on the right side represent 96 lambda DWDM configuration. Since each fiber has a multitude of DWDM lambdas the links are "approximate paths". When we increase the number of fibers a link, we increase the number of "approximate paths" for the link. If a pair of nodes are inter-connected with fibers with multiple DWDM lambda configurations, there will be a link per kind of DWDM lambda configuration. Each link represents the product of the number of lambdas in a DWDM configuration and the number of fibers between the nodes that have that DWDM configuration.

Figure 2 shows the same region of the network. Note that we have 2 different kinds of nodes in the figure. The larger 6 sided nodes have lambda translators available for assignment to form more complex routes. The circular nodes are source/destination nodes for routes, but do not have (or need for this example) lambda translation capability. The links in this graph are the cache entries of the lower level. For example the link between nodes 1 and 4 in Figure 2 are all the "approximate paths" that the route explorer discovered while exploring the lower level.

Figure 3 shows a sampling of the cache entries discovered for the "raw lambda and micro-mirror level of the network".

Figure 4 shows the possibilities that the route explorer discovered between nodes 1 and 9 at the "lambda translation level". Note the ***Tx*** in each possibility. It represents the lambda translator and its placement in the route. The route selection method at the "lambda translation level" selects a route that requires the least number of lambda translators (1 in this case) and for the set of possibilities that are the best, it chooses a subset that requires the fewest signal regenerators, and from that subset, the shortest path in fiber distance. This takes more exhaustive analysis (thus more computation) but it is performed on a small subset of the possible routes in the network. The first 3 routes are the combinations from the 1->4 and 4->9 links. The group of 2 routes are the combinations from the 1->5 and 5->9 links. The last single route is the combination of the 1->6 and the 6->9 links.

There is one constraint that exists at the lower levels of the network that are difficult to grasp at first exposure. Micro-mirrors or bubble switching of a lambda along the path of its route, don't change its optical properties other than introducing some loss of signal strength. That means a path that transits multiple fibers has to be assigned to the same lambda on each of the fibers along the path. But as lambdas are assigned and surrendered, over time there is a possibility that we can reach a situation that an exact path can not be selected from the "approximate paths" because although every link along a "possible path" has available lambdas for assignment, there is not one lambda that is available on all links of every possible path. This condition we will call "lambda assignment blocking". Similar situations with MPLS labels or ATM's VPI/VCI labels are why label

switching is done for each link transition along their paths. In this case, changing a label won't resolve the problem. Only the capability of switching the lambdas a signal is carried over along its path will solve the problem. That is a secondary reason to have "lambda translators". The invention anticipates this problem by including a metric used in route exploration called "lambda assignment availability". It summarizes the lambdas available over each link traversed in the exploration and will not continue to explore unless there is at least on lambda assignment shared by all possible links along the paths it is exploring. This creates the property that only paths with available end to end lambda assignments will be entered into the cache at the lowest level in the network. It also establishes the point at which we age out those lower level cache entries. For example, if the path in the "approximate path" set that has the least number of lambda assignments available has 22 of them, then after the 22nd route request after the cache was filled we will age it out. In other words, our cache age out mechanism assumes the worst case; that all 22 of those requests will be successful and will use a lambda assignment from the "approximate path" set that is the most critical.

Route exploration (i.e. cache fill), Route Selection, Approximate path ordering, and cache ageing all depend upon metrics. Each node and link has a set of properties that establish an their contributions to the metrics of paths that traverse them. Fibers are primitive links. They have insertion loss, DWDM characteristics (lambda counts and maximum data transfer rates), and lambda assignment profiles. Each level in the Graph has its own set of metrics that it uses. The metric sets consist of primitive metrics such as insertion loss and lambda assignment availability. The general algorithm defines primitive metrics as

classes with the same set of pure virtual methods. The specific algorithm defines primitive metrics that are instance objects of these classes. A metric set used at a specific level, is a metric set class instance with a defined set of primitive metrics and a set of pure virtual methods that are related to those of the primitive metrics.

Route exploration uses 4 pure virtual methods to generalize itself so that it is not dependent upon any specific metric. In this way it is different from Djikstra's algorithm that has a single metric; cost, that is embedded in the algorithm. Graph's route exploration algorithm is a generalization of the maze runner algorithm of first year computer science textbooks. It is modified in that it has isolated the metric from the algorithm and uses metrics as plug-ins to specialize its behavior and that it has a heuristic borrowed from autonomous intelligent agents. This in turn was modeled after the behavior of Bees and Ants. It is called the pheromone metric that is associated with every link in the level of the graph. Each probe in Route exploration will inspect the pheromone metric to see that it can improve on the probes that preceded it. If it can, it continues on after leaving its own pheromone values in the metric. This is the function of the improves and update methods of all metrics. The algorithm also tests its probes against the routing requirements metrics that rule out routes whose aggregate metric values exceed critical thresholds. This is the function of the meets_requirements method of the metrics. The step_over method is used to aggregate metric values along a path.

The pheromone and route requirement heuristics limit the growth of the algorithm to almost linear growth thus making it feasible to use the maze runner.

The invention, as disclosed, uses the same route explorer general algorithm with different metric sets for all levels. A variation of the invention could use different route explorers for each level. It would be possible to use Djikstra's algorithm for example in an additional level built above the highest level we have described and still be within the description of the invention.

The primitive metric types are: Boolean, additive, probability, minimum, set of minimums. Their names identify their aggregation property. An additive metric computes an aggregate property as the sum of the path added with the sum of the component being stepped over. Boolean aggregation is the Boolean AND of the two inputs. Probability deals with values between 0 and 1 and produces the multiplicand of the two inputs. Minimum produces the minimum of the two inputs. Set of minimums produces a new set whose result elements are minimums of their like input elements. All metrics have the property that they are continuous in the range they are used in and that they are monotonically asymptotic from a best value to a worst value. Probability is he easiest to recognize, its best value is 1 and the worst is zero. A value of 0.9999 is worse than 1 and better than 0.9998. A minimum has a best value of infinity and a worst of zero. Boolean has a best of 1 and a worst of 0. Set of minimums has a best of all elements being non-zero and a worst of all elements having zero values. Additive is best at zero and worst at infinity.

Specific metrics used for optical route selection are mapped to one of the 5 types described above. The table below defines the specific primitive optical metrics in column 1, the type it is an instance of in column 2, and its units in column 3.

Insertion Loss	Additive	Decibels
Lambda assignment availability	Sum of Minimums	Lambda indices
Latency	Additive	Milli-second
Jitter	Additive	Milli-second
Bandwidth availability	Minimum	Bits per Second
Bit Error Rate	Probability	Probability
Route Length	Additive	Meters
Regenerator Availability	Minimum	Integers
Lambda Translator Availability	Minimum	Integers

Sets of metrics are objects that have an ordered list of primitive metrics, a mapping of those primitives to its improves method, and a mapping of those primitives to its meets_requirements method. The ordering of the primitive metrics is used to establish

precedence for a canonical ordering that preserves 'Simplest Shortest First' within the scope that the metric set is defined within.

Whole Fiber level

- Ordered Primitives – Insertion loss, route length, minimum fiber availability, latency, jitter, bit error rate probability
- Improves map – 0, 1, 2
- Meets requirements map – 0,1,2

Path Selection – lambda configuration selection, lambda assignment selection, fiber selection in that order

Lambda level

- Ordered Primitives – Insertion loss, lambda assignment availability, route length, bandwidth availability, lambda translator availability, signal regenerator availability, latency, jitter, bit error rate probability
- Improves map – 0, 1, 2
- Meets_requirements map – 0,1,2
- Path Selection – lambda configuration selection, lambda assignment selection, fiber selection in that order

Note – the primitives after route length are used for precedence, but level 5 adds no significant aggregation contribution to these metrics. Only the first 3 metrics are involved in improves and meets_requirements.

Signal Regenerator level

- Ordered Primitives – Lambda assignment availability, route length, bandwidth availability, lambda translator availability, signal regenerator availability, latency, jitter, bit error rate probability
- Improves map – 0, 4, 1
- Meets_requirements map – 0, 4, 1
- Path Selection – Least signal regenerators, largest lambda assignment availability, shortest route length – Lambda picks best path from selected approximate path

Lambda Translator level

- Ordered Primitives – route length, bandwidth availability, lambda translator availability, signal regenerator availability, latency, jitter, bit error rate probability
- Improves map – 0, 2, 3
- Meets_requirements map – 0, 2, 3
- Path Selection – Least lambda translators, least signal regenerators, shortest route length – Signal regenerator level picks best path from selected approximate path

O-E-O level

- Ordered Primitives – route length, bandwidth availability, lambda translator availability, signal regenerator availability, latency, jitter, bit error rate probability

- Improves map – 0, 1, 2, 3, 4, 5, 6
- Meets_requirements map – 0, 1, 2, 3, 4, 5, 6
- Path Selection – Largest available bandwidth availability, least lambda translators, least signal regenerators, shortest route length – Lambda translator level picks best path from selected approximate path

Multiplexor level

- Ordered Primitives – route length, bandwidth availability, lambda translator availability, signal regenerator availability, latency, jitter, bit error rate probability
- Improves map – 0, 1, 2, 3, 4, 5, 6
- Meets_requirements map – 0, 1, 2, 3, 4, 5, 6
- Path Selection - Largest available bandwidth availability, least lambda translators, least signal regenerators, shortest route length – O-E-O level picks best path from selected approximate path

The graph object has many external interfaces and methods that are used to download the networks mesh definition and changes to it including periodic updates of nodes and links current resource availability. These will not all be described here because there is nothing unique in the invention's implementation of these methods. We will constrain our description to the portions that are unique to the invention, which is its path determination method(s). The best embodiment of the invention is written to be a plug in

to provisioning and OSS systems of network service providers. The description of how it interfaces with these systems is not unique and is considered to be more germane to copyright protection than a patent and will not be included in this application. In other words, we will try to limit ourselves to a description of what is unique and advantageous to keep what will be a long complex description from becoming even longer.

The graph object uses objects:

a. Node
b. Node_level
c. Link
d. Link_level
e. Cache
f. Cache_level
g. Primitive_metric_types
h. Metric_sets
i. Path
j. Exact_path
k. Approximate_path
l. Path_request_parameters
m. Path_result_parameters

Graph maintains multiple maps (i.e. neighbor tables). One at each level configured into the graph. The embodiment described uses 5 levels but the graph as implemented can easily be configured to more or fewer levels.

Nodes can be added, deleted, enabled_forwarding, enabled_routing, disabled_forwarding, disabled_routing, Node_update, port_added, port_deleted, port_enabled_forwarding, port_enabled_routing, port_disabled_forwarding, port_enabled_routing, port_updated. Disabling routing allows a port or node to continue forwarding maintaining current connections but allowing no new routes to be established that transit the Disabled component. Based upon the properties of the Node, various node_level objects will be derived from the Node base class and entered into the various data models for the levels that the graph utilizes.

Links can be added, deleted, enabled_forwarding, enabled_routing, disabled_forwarding, disabled_routing, Link_update, lambda_added, lambda_deleted, lambda_enabled_forwarding, lambda_enabled_routing, lambda_disabled_forwarding, lambda_enabled_routing, lambda_updated. Disabling routing allows a lambda or link to continue forwarding maintaining current connections but allowing no new routes to be established that transit the Disabled component. Based upon the properties of the link, various link level objects will be derived from the Link base class and entered into the various data models for the levels that the graph utilizes.

The data structures needed to understand the invention are:

a. A directory of connections that have been established for the network.

b. An inventory of fibers and metallic cables used as links

c. An inventory of transmission equipment used as link/node interfaces

d. An inventory of Interconnection Nodes (primarily switches or patch panels)

e. A structured list of cache objects; each of which have the set of objects that make up a level of the graph

f. A list of lambda configuration structures that are derived from the inventory of transmission equipment

Each cache level has the following components:

a. A map of the interconnections at this level (neighbor list)

b. A cache of paths between pairs of nodes that have common routing characteristics

c. A set of primitive metrics that are used to route paths at this level

d. A cache fill method specific to the level

e. A path commit method specific to the level

f. Set of cache sub-keys

The path commit method uses node and link methods that may optionally command device specific command plug-ins to establish the desired circuits.

The interconnections map is a directory. Each entry has a key that is the identifier of a node that participates at this level in the graph. The values are lists of tuples; a link and a node. The node is the other end of a link that the key node transmits to. This allows the interconnection between 2 nodes by multiple links of varying routing characteristics.

Each cache's path store is a directory that is keyed at the top level with a tuple; the source node and a routing characteristic identifier. The entries are directories whose keys are the destination node identifiers and whose entries are ordered lists of approximate paths.

Each level has a metric set that is specific to the level. Each set is made up of an arbitrary number of primitive metrics; each of which is one of 5 basic types. Each set has an ordering precedence among its primitive metrics so that comparing 2 instances of the sets will establish either an equality or precedence ordering. Each primitive metric has the following methods: step_over(), meets_requirements(), improves(), update(), is_equal(), better().